DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2019

Automated Triangular Arbitrage: A Trading Algorithm for Foreign Exchange on a Cryptocurrency Market

Sanghyun Bai
Fred Robinson

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

This project uses software development to investigate the link between software and finance. The focus of the work is developing and implementing a trading algorithm which seeks to make profit by making trades based on arbitrage opportunities between currencies. Specifically, the sets of currencies examined are two fiat currencies and one cryptocurrency. Trades are made by combining a blockchain system, which maintains the cryptocurrency, and the live foreign exchange market, which enables fiat currency exchange. The main methodologies for carrying out the research are test-driven development and the use of a simulation to facilitate trades. By passing all of the unit tests, the software is verified. In addition, data gathered during runs of the simulation show that the algorithm successfully identifies arbitrage opportunities and turns a profit on average over many runs. This project proposes an interesting topic for further research in the field of blockchain technology used for financial trading.

Keywords

blockchain, cryptocurrency, arbitrage, foreign exchange, trading algorithm, optimization, java

Abstract

Detta projekt bygger på mjukvaruutveckling för att undersöka kopplingen mellan programvara och finans. Arbetet fokuserar på att utveckla och implementera en algoritm för valutahandel som försöker skapa vinst genom att genomföra handel baserade på arbitragemöjligheter mellan valutor. Specifikt är de uppsättningar valutakurserna som undersöks två ordinarie valutor och en kryptovaluta. Handel utförs genom att kombinera ett s.k. blockchain-system, som upprätthåller kryptovalutan, och den ordinarie valutamarknaden för vanlig valutaväxling. De viktigaste metoderna för att genomföra undersökningen är testdriven utveckling och användande av simulering för att genomföra valutahandel. Mjukvaran verifieras med hjälp av en uppsättning enhetstester. Dessutom visar datan som samlats under simuleringar att algoritmen framgångsrikt identifierar arbitragemöjligheter och i genomsnitt ger en vinst över många körningar. Detta projekt utgör på så sätt ett intressant ämne för vidare forskning inom blockchain-teknik för finansiell handel.

Authors

Fred Robinson <fredrob@kth.se>
Sanghyun Bai <sanghyun@kth.se>
Information and Communication Technology
KTH Royal Institute of Technology

Examiner

Markus Hidell
Division of Communication Systems
KTH Royal Institute of Technology

Supervisor

Peter Sjödin
Division of Communication Systems
KTH Royal Institute of Technology

Contents

1 Introduction

This project focuses on the practice of software development to research the use of financial trading algorithms on a blockchain system. The blockchain system is a distributed ledger of records that is transparent to all and keeps the details of transactions between two parties "efficiently and in a verifiable and permanent way."[1] One of the most common uses of blockchain technology is to develop a cryptocurrency, or virtual currency which users can make transactions online with. These transactions are verified by the blockchain.[1] The project's sponsor, Centiglobe, uses their proprietary blockchain software to establish a cryptocurrency.[2] Using this cryptocurrency and the blockchain system as a basis, this project implements the trading concept of triangular arbitrage into software.

This trading strategy focuses on the foreign exchange market, where trades are made between different currencies and commodities. Triangular arbitrage seeks to find price discrepancies between two currencies by trading through a third currency. These discrepancies can lead to profits if investors make the right trades. While triangular arbitrage is a well known strategy already, this thesis seeks to further develop it by using a software implementation of the strategy to trade between two fiat currencies and one cryptocurrency. The software implementation of the triangular arbitrage strategy is referred to as the automated arbitrage algorithm.

To implement the trading algorithm on the blockchain platform, common software development practices are followed. The algorithm is tested using a simulation. The simulation approximates a real market by using live market pricing data and using simulated traders to make trades amongst each other. The practice of test-driven development is used to modularize the system as much as possible and allow for testing of individual components.

1.1 Problem

The project aims to investigate two main topics in algorithmic trading on a blockchain system. Firstly, proving that arbitrage exists between two fiat

currencies and one cryptocurrency. Secondly, showing that profit can be made from trading based on this arbitrage. Both of these goals are accomplished by developing and implementing algorithms for the blockchain system. The problem can be separated into the financial components and the technical components associated with completing the two topics.

The financial side of the work is simple in the sense that the sole objective of the work is to capture profit. This is done by trading on an emerging technology platform with new currencies. The algorithm which is implemented is a further development of an existing trading strategy; triangular arbitrage. So far, this strategy has only seen significant use between three fiat currencies or three cryptocurrencies, but not in a mixture of the two currency types. Furthermore, the technical side of the work has many components that must be addressed in order to develop the trading strategy and system. Firstly, a software platform needs to be developed that is connected to and can gather data from two different financial markets; the cryptocurrency market and the real market. Secondly, it is essential to create an algorithm that detects arbitrage between the two markets and tries to make a profit by trading based on that opportunity. The final challenge is further developing the sponsor's proprietary simulation program to be a robust software system and be able to run the automated arbitrage algorithm.

The financial components are difficult because all of the analysis is done using virtual money in a simulation rather than real money on a real market. In addition, the simulation is not a perfect replica of the real market; traders in the simulation make their trading decisions according to the code, rather than by following economic trends and market behaviour.

The challenges mainly exist on the technical parts of the project, however. The first challenge is creating a pricing strategy. This is done to increase the accuracy of the trading simulation by representing the real markets and giving the traders in the simulation accurate prices for their transactions. The pricing strategy thereby helps to accomplish the first goal of showing arbitrage. The implementation of the pricing strategy includes querying the cryptocurrency forex market to receive pricing data. This is done through API calls to the crypto market. Then, to determine the prices used by the traders in the simulation, the pricing data from the crypto market is gathered and coalesced to form a normal

distribution from which the traders receive a price value.

The next major challenge is developing the arbitrage trading algorithm. This includes connecting the two different markets through using two different APIs. This means studying the API supplied by the live foreign exchange market and the API used in the pricing strategy; the one from the blockchain that allows communication with the blockchain system for cryptocurrency transactions. In some cases, it is also necessary to not only make the appropriate API calls, but to develop the API for the blockchain system to facilitate specific functions that were not previously implemented. The algorithm is completed by modularizing the trading strategy into separate parts that are run sequentially. The second part of the algorithm can run if and only if the first part has completed successfully.

The final major technical challenge in the project is to further develop the given trading simulation in a robust manner. The simulation is developed to be able to run using the implemented pricing strategy and the arbitrage trading strategy. To do this, a new type of virtual trader must be implemented to run in the given simulation. This trader will execute trades based on the arbitrage strategy. All traders in the simulation will use the developed pricing strategy. The software system is developed in a robust way meaning that there are as few 'hard programmed' solutions to problems as possible and that the simulation can be run with the specifications for the arbitrage trader without having to change any of the source code.

1.2 Purpose

The purpose of the thesis work is to advance the main research topic; trading algorithms for a new global blockchain currency. The trading algorithm itself, triangular arbitrage, is already well known and has many documented examples of use in foreign exchange markets. The new area of interest with this work is implementing the trading strategy to allows for trades between two different currency types; fiat currencies and cryptocurrencies. This algorithm is crucial to prove, using software, that there is arbitrage and therefore, profitable opportunities for trading between the two markets. The end product should be usable by an external sponsor to present it to stakeholders or gateways.

Developing the trading algorithm is also valuable work since it contributes to the field of blockchain systems, an emerging technology. While there are many examples of work in financial technology where developers have created software systems using APIs to traditional live markets, there is room for much more research into blockchain markets. This thesis seeks to contribute to the field of research that combines robust software systems using blockchain technology and the financial industry.

1.3 Goal

The goal of this project is to provide an upgraded, robust piece of software that acts as a trading simulator capable of using the developed arbitrage trading strategy. The software should be further developed from the given simulation by the sponsor. It should be able to accept actors in the simulation other than individual gateways who make trades based on generated trading intentions. The final product - the given simulation combined with our pricing and trading strategies - should simulate a real market and perform trades between multiple currencies with one actor on the market using the developed arbitrage trading strategy. Therefore, the arbitrage trader should be able to accumulate, process, and analyze real-time data from the live forex market and the cryptocurrency market on the blockchain concurrently by using appropriate API calls. If applicable, the arbitrage actor should attempt to create a profit by making trades.

1.4 Methodology

To perform the work, the main research methodology chosen is test-driven development of simulation software. Since the proprietary blockchain system is still in development, the trading algorithm cannot be tested on a live market. Therefore, testing occurs via a simulation that creates a realistic market environment. Theoretically, the algorithm could be tested (and would produce more accurate results) on the real market. This could be done in the future when the blockchain system is fully developed.

During the development process of all components of the software system,

the concept of test-driven development is used. This means writing individual tests for each component of the system which correspond to the requirements, and then writing the components to pass the tests.[3] This methodology is beneficial in that it allows development only on the parts that are necessary to meet the project requirements. With this project, there are many areas that could be improved further and further, iteratively, if time permits, but since the work is completed within a limited time scope, test-driven development is chosen to focus the effort on the necessary components only.

1.5 Delimitations

To meet the goal of producing a useful, robust piece of software and to keep the amount of work within reasonable bounds for the period of time there is to develop, certain limitations are made for the project. Firstly, the work is completed assuming that the sponsor's blockchain and cryptocurrency system will work as promised, even if they are still in development at the moment. Namely, there are no fees for trading or cancelling orders on the cryptocurrency market. Also, there is no trading fee on the live forex market. Furthermore, the project is delimited by not taking into account the transfer between fiat currencies and their virtual counterparts (e.g. USD to BitUSD); it is assumed that 1 unit of fiat currency is equal to 1 unit of the virtual counterpart.

Since the focus of the project and trading algorithm is on foreign exchange, it is assumed that any user of the algorithm has the appropriate personal infrastructure to use it. That means that the user needs bank accounts, or other equivalent systems to hold and trade foreign currencies. It is assumed that this infrastructure is in place and foreign currencies can be traded freely, without fees or restrictions, and there is ample liquidity in each of the foreign currencies so that they will always be available.

1.6 Benefits, Ethics and Sustainability

Anytime one is dealing with money and transactions, there are ethical considerations they must make. When referring to the foreign exchange market,

some people use the term 'zero sum game.' This describes the concept that, for all traders who make a profit on the market, there must also be traders who make an equally large loss; i.e. the total sum of profits and losses is zero across all traders. This same concept applies whether one makes transactions using cryptocurrencies or fiat currencies. It is up to the individual whether they are comfortable with profiting only when others are losing.

Foreign exchange is currently dependent on centralized institutions to maintain cash flows and liquidity to facilitate trading. This can be problematic if the responsibility is only left to a few large actors, such as banks. Rama Attreya documented an example of this in a Boston University financial law review: "As of May 2015, the Department of Justice (DOJ) reported that five key banks: JPMorgan Chase & Co., Citicorp, Barclays PLC, The Royal Bank of Scotland (RBS) PLC, and UBS AG had pleaded guilty to felony charges." [4] These five banks were charged with colluding with each other to manipulate exchange rates on the forex spot market.

In a cryptocurrency market however, this is not the case. One of the main advantages of using a cryptocurrency is decentralized exchange. The problem of shortage of liquidity in this market is solved by allowing any individual to be a market maker. With our company's proprietary cryptocurrency system, any individual is able to be a market maker. This individual freedom combined with the transparency of transactions provided by using a blockchain system allows for sustainable development. Overall, the cryptocurrency system allows for individual economic growth, which contributes to economic sustainability, and for a more transparent market, which promotes social sustainability.

1.7 Outline

The thesis is broken down into sections which cover each area of the work. Some of the sections are broken down further into subsections. Chapter two discusses the background information and related works to the topic. There is a description of necessary technical and financial information for the reader to understand before they continue with the thesis. Also, related research projects are discussed.

In chapter three of the thesis the methodology and methods used for the work

are discussed. This section provides the plan for how the results will be generated after the execution of the simulation.

Chapter four consists of design decisions for each of the pieces of software; these are discussed and motivated. In addition, the implementation process for each part of the project is explained.

In chapter five, the results are presented and analyzed according to the methodologies laid out in chapter three. There is a discussion given on the results.

The last chapter of the thesis, chapter six, gives conclusions on the research and also provides a look to the future for how the work can be continued.

2 Background and Related Works

The background gives the reader necessary information regarding the rest of the thesis. In order to fully understand the problem description, the solution designed for it, and the methodology used to carry out the work, a foundation of knowledge is provided. There is a description on the technical aspects of the work: blockchain and algorithmic trading. Also, a description of the financial side of the work is given on foreign exchange and market makers. Finally, the background is concluded with a section on related works: research that has been performed which is similar to this project.

2.1 Blockchain

2.1.1 General Description

Unlike a traditional database, a blockchain is a so-called distributed database; there are many copies across many nodes. The database is an open ledger of transactions. The system is considered decentralized because each node has its own copy of the entire blockchain, but there is no one particular central server. To validate or transfer data, a blockchain uses peer-to-peer (P2P) communication. To provide for secure communication, each copy of the distributed database is encrypted by hash-functions. In addition, all of the transactions are "append only," meaning that blocks can only be added to the chain, but not deleted. When the chain is updated in any way, all changes are reflected on all copies of the databases hosted on the different nodes. The result is that the database cannot be altered or changed. Also, every entry is permanent in the database. The structure of the whole system works by each transaction being sequentially added to the database in a 'block,' and together they make the 'blockchain.'

On a blockchain system, transactions are handled differently than a traditional client-server network. This is because of the use of a distributed database that runs on a P2P network. In a paper from Deloitte, Richard Bradley discusses how the use of a blockchain system over a traditional system using an intermediary is beneficial: "The intermediary is replaced by the collective

verification of the ecosystem offering a huge degree of traceability, security and speed."[5] A relevant example of this is a financial transaction, where with the use of blockchain technology, money could be sent directly from one person to the other. This offers the 'traceability, security and speed' discussed by Bradley, as opposed to a traditional system, where money needs to be sent from one person to another through a bank or other financial institution. These two transaction types are illustrated below in Figure 2.1.

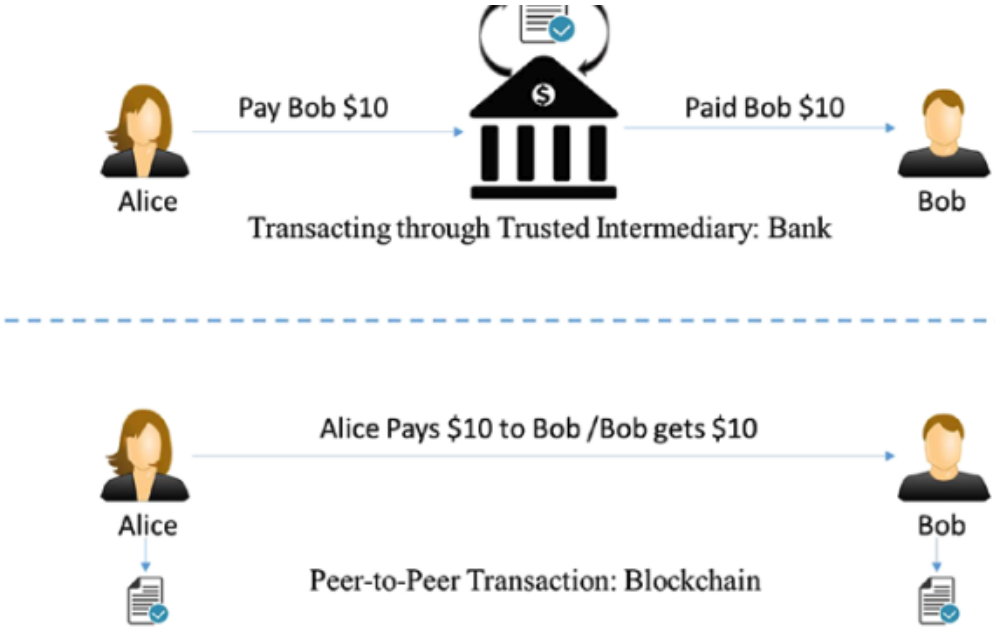

Figure 2.1: *Transaction without blockchain vs transaction with blockchain [6]*

The use of blockchain makes transactions safe and transparent, which is particularly important for money transfer. Bitcoin is an example of a way to transfer money by using a blockchain system.[7] Just like some other cryptocurrencies, Bitcoin facilitates safe money transfer without a bank or other third party. In a traditional wire transfer, for example, the bank verifies, secures and settles the transaction.[6] With the use of a cryptocurrency such as Bitcoin, the middleman is removed since the system is decentralized and therefore the transaction is verified by each of the nodes that comprise the system. It is then appended to the blockchain; this ensures the verification and settlement of the transaction. The safety of the transaction is guaranteed by the encryption that is a part of the structure of the blockchain.

2.1.2 Technical Description

The layering of the different functions of the blockchain help the reader to understand the complex system. The explicit, modularized structure allows the identification and correlation of various sections. Also, the modules ensure simplicity and ease when updating the system. As of now, there are no agreed global standards for structuring the layers of blockchain. However, the book ”Beginning Blockchain,” written by Singhal, describes blockchain in five layers. It consists of the Application, Execution, Semantic, Propagation, and Consensus Layer.[6] It is important to remember that blockchain technology is established on top of the internet. This means the five layered blockchain survives on the highest layer (application) of the Internet protocol stack.

The blockchain application layer provides the desired functionality to the end user. It can be implemented in the traditional way of software development which is based on the client-server model. For instance, to use a web browser as a client, it is required to retrieve a web page from a web server. Although it is possible to implement, “Ideally, good blockchain applications do not have a client-server model, and there are no centralized servers that the clients access”[6] Instead, the peer-to-peer model is used in order to create a decentralized system. In the peer-to-peer system, arbitrary end systems communicate to each other directly. For instance, a user in the network requests service from peers and provides service in return to them. This provides the characteristic of self scalability meaning new peers increase service capacity as well as service demand.

The execution layer handles the set of instructions defined by the application layer. These instructions must be executed in every node in the blockchain network independently to ensure the correct execution of the transaction. Moreover, execution of instructions with identical parameters must always generate the identical output on every node. This serves to avoid inconsistencies.[6] More importantly, the instructions to execute need to be in the form of a ”smart contract”. The term smart contract was coined by Nick Szabo. He describes, ”A smart contract is a set of promises, specified in the digital form, including protocols within which the parties perform on these promises.”[8] In blockchain, every node can create a smart contract which functions as an

immutable agreement. The advantages of the smart contract are that the system does not need a third party to verify the transactions. Also, the record of transactions is public; it is accessible to all nodes so it cannot be lost.

The semantic layer structures transactions and blocks in a logical manner. This layer validates the instructions that will be executed by the execution layer. Validation must be performed in every node. It can be accomplished by traversing previous transactions. Furthermore, the semantic layer defines the rules of the system. The rules contain, for example: data structures, data models, and storage modes. Also, the layer defines how the blocks are linked with each other. Every block in the network of the blockchain carries the hash information of the block before it.[6]

The book ”Beginning Blockchain” explains, ”the Propagation Layer is the peer-to-peer communication layer that allows the nodes to discover each other, and talk and sync with each other with respect to the current state of the network, So, transaction/block propagation in the network is defined in this layer, which ensures the stability of the whole network”[6] The propagation process acts similarly to broadcasting in the Internet protocol; when a new block is proposed by a user, it broadcast to the entire blockchain network.

The central objective of the consensus layer is that every node agrees on one consistent state of the ledger. This layer, therefore, assures safety and security of the blockchain. Examples of consensus protocols are Proof of Work (PoW), Proof of Stake (PoS), delegated PoS (dPoS) and Practical Byzantine Fault Tolerance (PBFT).[6]

2.2 Foreign Exchange

2.2.1 Foreign Exchange Market, Rates and Currencies

Foreign exchange (FX or forex) trading involves the exchange from one currency to another. This is done on a global market; the largest market in the world as measured by liquidity. According to the Bank for International Settlements, the average daily trading volume of the global foreign exchange market in 2016 was over 5 trillion USD.[9] Transactions on the FX market occur between two traders

who wish to exchange currencies. These currencies, most often official forms of money from sovereign nations, but sometimes other resources (such as gold and silver) are represented on the market with a three letter code. For example, the US dollar is denoted "USD," the British pound "GBP," the euro "EUR," and so on. A foreign exchange rate, for example USD/EUR, tells the trader how much of one resource is equivalent in value to one unit of another resource; in this case, how many US dollars equals the value of one euro. Many factors affect the exchange rate between two currencies at any given time, including things like the GDP of the nations issuing the currencies and the purchasing power parity between the two nations as well. Kenneth Rogoff of Princeton University explains this concept in his 1996 article in the Journal of Economic Literature, "The Purchasing Power Parity Puzzle": "The basic building block for any variation of purchasing power parity is the so-called 'law of one price' (LOP). The law of one price states that for any good *i:* $Pi = EPi^*$ where Pi is the domestic-currency price of good i, Pi^* is the foreign currency price, and E is the exchange rate, defined as the home-currency price of foreign currency."[10] In the given equation, E is dependent on the price of a certain good in two different currencies, so the exchange rate can be affected based on the difference in purchasing power between two currencies; the purchasing power parity. Many other factors affect the exchange rate between any two currencies, but this thesis will not go into further detail on this topic.

In today's society, as globalization occurs on a larger scale each day, the foreign exchange market includes a vast number of currencies. Adam Kritzer estimates there are "perhaps 160 different currencies in the world" in his book on foreign exchange.[11] Of these worldly currencies however, only 17 of them, the so-called major currencies, account for over 90% of the volume of global foreign exchange volume.[11] Kritzer also explains that over 97% of the same global volume is captured by the major currencies plus the so-called exotic currencies, which are the next 20 most traded currencies.[11] Although thousands of possible currency pairs exist with the 160 total currencies in the world, trading between two uncommon currencies usually occurs with a major currency as the 'middle-man.'

Imad Moosa explains the concept of this so-called triangular arbitrage in his text on quantitative finance: "the cross exchange rate between two currencies

calculated from their exchange rates against a third currency or a numeraire must be identical to the cross rate that is actually quoted."[12] For example, in the foreign exchange market, if a trader is looking to convert from one exotic currency to another, he will most likely need to trade in two steps using a major currency. This is illustrated in figure 2.2.

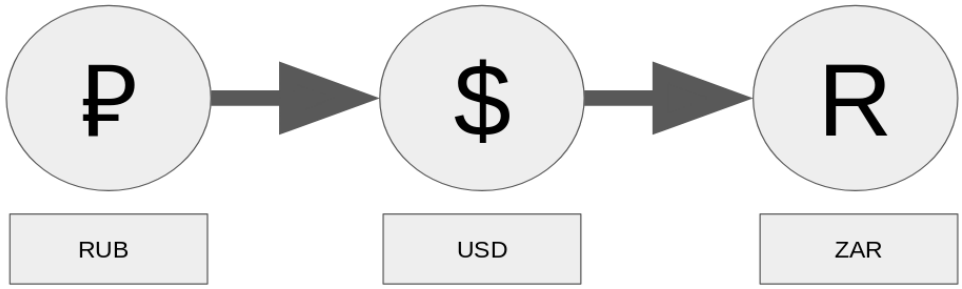

Figure 2.2: *Exchanging two Exotic currencies*

Here, a trader looking to convert Russian Ruble into South African Rand. Since this currency pair (ZAR/RUB) is uncommon, the trader will need use the the US dollar (USD) to exchange. In the first step, the trader will need to buy US dollars with his Russian Rubles (USD/RUB). Next, he can purchase South African Rand with his US dollars (ZAR/USD). The ultimate result is the same (ZAR/RUB), but the purchase and sale of US dollars in between is necessary since the trader likely cannot buy Rand directly with his Rubles. As Moosa states, in a perfect market, the trader should receive an identical amount of ZAR for his given RUB, whether or not he buys and sells USD in between. A difference in the exchange rates in either of the two steps of this conversion is what leads to arbitrage and that is where traders can profit; they exploit the difference in their favor.

2.2.2 Market Makers and Profit

There are many roles for the different actors on the foreign exchange market (or any market for that matter), one of these being market makers. Market makers, also called liquidity providers, promote trading on the market by buying and

selling most any currencies from individuals, or other firms. The US Securities and Exchange Commission (SEC) describes market makers on the forex market: "Market makers take the opposite side of any transaction" This occurs because, as the SEC explains, there is no "central marketplace" in the forex market. Individual traders, therefore, go to market makers in order to buy or sell the currencies they want to.[13] In this way, market makers provide liquidity to the market; they have cash and currency at the ready to buy and sell. They also earn a profit since they may give traders prices that are favorable to themselves based on information that is difficult for individual traders to obtain.

Profit for market makers (and other brokers) on the market is determined largely by arbitrage. The arbitrage is exploited by the brokers to earn profit through the prices that they offer clients. For an individual trader, the price that a currency can currently be sold at is referred to as the bid price. The price that a currency can currently be bought at is the ask price. A dealer uses these to form the bid/ask spread (or just spread) for each currency that it deals in. The spread is calculated as the difference between the bid price and ask price; the spread is the amount of money that the market maker earns for providing its service to the market (i.e. liquidity).

2.3 Algorithmic Trading

Algorithmic trading is a method to trade on a financial market with the assistance of algorithms. In his book, "Successful Algorithmic Trading," Halls-Moore describes the topic as trades "which are executed in a predetermined manner via an algorithm specifically without any human intervention." He stresses that the condition "without any human intervention" is crucial; one of the main advantages of trading using algorithms is that they do not require any discretionary input from the trader.[14] Algorithmic trading can also be automated so that they run without constant direction from the user.

In general, trading describes the exchange of one resource or currency for another. To use a market and perform a transaction, a trader places an order. Two major types of orders used here are the market orders and limit orders. In his book, "Successful Algorithmic Trading," Halls-Moore describes a market order

as one that "executes a trade immediately, irrespective of available prices." [14]. So long as there are buyers and sellers, a market order will always be filled. A limit order, on the other hand, allows the trader "to determine the worst price at which the trade will get executed, with the caveat that the trade may not get filled partially or fully." [14] As opposed to a market order, which is always executed if there are willing traders, a limit order will only be executed if the price is above (sell limit order) or below (buy limit order) the defined amount.

This thesis will focus on trading in two different financial markets: the foreign exchange market and the cryptocurrency market.

2.4 Related Work

Studies from Dong and Dong in "Bitcoin: Exchange rate parity, risk premium, and arbitrage stickiness." and Pieters and Vivanco in "Financial regulations and price inconsistencies across Bitcoin markets." prove significant discrepancies in Bitcoin prices across different exchanges.[15][16] Because we know the real foreign exchange market to be inefficient and the studies on Bitcoin show the cryptocurrency market to be inefficient as well, we expect to see inefficiencies when trading between the two markets. These inefficiencies mean that there are opportunities to exploit price differences in the two markets through triangular arbitrage and thus earn a profit.

In a research paper, "Triangular No arbitrage Estimation through Bitcoin: An Application in Venezuelan Bolivars," Wang, Hsu, and Chen try to estimate the exchange rate for Venezuelan Bolivars using Bitcoin. They assume "triangular no-arbitrage;" there is no arbitrage to exploit in the market since there are no inefficiencies in the market.[17] Although we do not make this assumption in our research, we use a valuable conclusion of theirs: "in comparing the conventional exchange markets, we conclude that trading through the Bitcoin market has higher transaction costs or requires higher risk compensation in general."[17] Although our research will be examining markets that do show inefficiencies and thus do have arbitrage opportunities, the conclusion noted by Wang, Hsu, and Chen applies whether there is arbitrage or not. Since part of our trading includes the cryptocurrency market, our returns must be higher than the transaction costs

or costs associated with risk compensation in order to make a profit.

A similar study to ours was performed by Reynolds, Sögner, Wagner, Wied: "Deviations from Triangular Arbitrage Parity in Foreign Exchange and Bitcoin Markets." They examine the occurrence of triangular arbitrage between some combinations of three fiat currencies and of two fiat currencies and Bitcoin. They state that: "Overall, our results confirm the relative rarity of triangular arbitrage within the market for fiat currencies, while highlighting the potential for financial market dislocations on the newer market for cryptocurrencies."[18]. By "financial market dislocations," the authors indicate opportunities for arbitrage. This study, therefore, directly supports our research since our algorithm trades based on a triangular arbitrage strategy across the real and crypto markets. One key difference between this study and ours is that we will use a proprietary cryptocurrency supplied by our external sponsor rather than Bitcoin.

3 Methodology and Methods

To carry out this research, the methodology chosen is to construct a model of the real market (i.e. the simulation) upon which the arbitrage trading algorithm can be tested and used to determine if the goals identified in the introduction are met or not. The software system is tested to see if it successfully implements the trading algorithm. In addition, the system is tested to see if the updated trading simulation can successfully run with an actor who uses the arbitrage trading algorithm.

The research process is explained, detailing how the whole project is carried out. Next, the data collected to test and validate the work is highlighted. The experimental design section explains how the work is carried out in terms of the setup and the hardware and software tools used. After that, the assessment of the validity of the data collected is given. The planned data analysis that shows how the data is used to produce the results is given. Finally, the evaluation framework is given; the detailed description of how the success of the project is determined.

3.1 Research Process

The research process describes how the technical work for this entire project is carried out. Throughout all parts of the implementation, the practice of test-driven development is used. Therefore, the first step to carry out the implementation is to write the test programs for features and functionalities of the algorithm. Next, the software is developed to pass those tests. This process is repeated iteratively until the entire system is built and passes all the requisite tests. By this point, each individual function and feature of the system (algorithm and simulation) is validated.

The next step in the research process is to collect data. The exact data collected is further specified the data collection subsection. Finally, the actor's account balance is monitored throughout runs of the simulation and, in combination with the unit tests for functions, the validity of the software is ensured. By showing that each function passes its test and the account balance

for the arbitrage actor is correct at each step of the algorithm, the implementation is thereby verified.

In addition to the testing and verification of the software system and implementation as a whole, the algorithm itself and the performance of the algorithm are also briefly discussed. Appropriate data are collected (e.g. the profit or loss made by the actor) and analyzed to determine if the algorithm is successful as a trading strategy and if it could generate profits for the user.

3.2 Test Bed

To perform the research, all implementation is done on the same computer using the same software. The specifics are as follows.

- Operating system: Ubuntu 18.04.2 LTS
- Linux kernel version: 4.15.0-46-generic
- Processor: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
- Java version: 11.0.2
- IDE and version: IntelliJ 2018.3.4

The main software used in the project is Java for implementation and the code to run the blockchain and the command-line wallet locally on the development machine. The blockchain and command-line wallet source code is proprietary information of the sponsor. The API to the live forex market (java library oanda v20 version 3.0.25) is used. Also, the following java libraries are used: commons-math3, gson version 2.8.2, json version 2018.1.30, apache version 4.5.3, and junit version 4.12. Finally, Excel is used to generate graphs for the analysis on the results.

3.3 Unit Tests for Software Verification

To verify the functionality of the software system, each piece of the system is tested individually. This method of testing each piece of the program individually is

particularly useful because the system as a whole is complex. Modularizing the testing process allows for quicker development, as less time is spent debugging, and allows for verification of the whole system if each individual part is verified through testing. Firstly, the pricing strategy and the random number generation system are tested. Then the algorithm is tested by individually testing the following: initialization of the actor, functions used in the algorithm, fetching data from the live forex market, step 1, and step 2 of the algorithm. A detailed description of each of these components is given in chapter 4 of the thesis. It may behoove the reader to read that chapter first in order to gain a better understanding of each part of the algorithm and then return to this subsection.

Testing the pricing strategy consists of making an API call to the external cryptocurrency market and ensuring the value by checking with the website at the same time. Since the pricing strategy also uses a normal distribution to sample from, the random generator in the selected java library function is tested by making 50 calls to the sample function with 5 different seed values and ensuring the sequences are all different.

To test the initialization of the actor who uses the arbitrage algorithm and other components for the simulation, a test function is written. Firstly, 3 gateways are initialized who each generate their own currency; this is tested. The arbitrage actor is then initialized and tested to ensure he has the correct initial balance: one million units of currency A, four million units of core, and zero units of currency B and C. The testing of the initialization is concluded with 5 orders being manually generated and then asserting the order book displays them correctly.

The individual functions that are used during the course of the algorithm are tested individually as well. Firstly, a function to identify the lowest ask order in an order book is tested; verification is given since the only orders in the order book are manually generated, so the order with the lowest rate is known. Next, the price fetching from the live forex market is tested by ensuring the ID and tokens used as credentials for the external system are valid. Then, the function makes the API call and prints the values retrieved to the console; they are checked with the website to ensure that the correct value is retrieved. After this, one of the main functions of the algorithm is tested - the *updatePriceRate* function. This function is tested by first storing the current values of the price rates: target to base, base

to core, and target to core. Then, a new ask order is manually generated which is lower than the current lowest. The function is called and each value is checked to make sure it now holds the updated price rates. At the same time, a test is run to ensure the *determinePriceRate* function works by comparing the returned value with a value calculated by hand according to the formula for the price rate (given in section 4.2.3).

After all the individual functions are tested, step 1 of the algorithm is tested. First, the test asserts that the initial balances of the gateways and the actor are correct (same as initialization test). Next, a subscription to the market is made so that notifications are received when there is an update to the order book. The initial order book is asserted to make sure that there exists only the manually generated orders. Next, the boolean that denotes whether or not a transaction is currently active or not is checked before step 1 continues. The step 1 function is tested by calling it and ensuring that the proper ask value is calculated and the order book is updated. This also tests to ensure that the algorithm places the ask order at the same rate as the current lowest ask in the order book, since this value is higher than the calculated ask rate (the motivation for this is explained further in section 4.2.3). The order book is printed to ensure the new order is there. The data collection portion of step 1 is tested by printing out the values of the transaction object to ensure they hold the appropriate values. There are three alternatives for step 1 after the ask order is generated. In the first case, the order is never filled and times out. This is tested by allowing the order to time out and ensuring that it no longer appears in the order book. In the second alternative, a bid order is manually generated at the same rate as the ask, but at a lower volume. This leads to the ask order being partially filled. In this case, the test function asserts that the order is partially filled by printing the order book and displaying the order with the correct value in the "% filled" field. In the third alternative, a bid order is manually generated so the ask order from the arbitrage actor is purchased. The callback function from the market subscription is tested here by ensuring it is called when the actor's order is filled. Finally, the actor's account balance and the order book are printed once more to ensure that they correctly reflect the purchase of the order.

The final unit test ensures the functionality of step 2 of the algorithm. To

begin, the initial balances of the gateways and the actor are tested by calculating the balances as the values from the setup; this test is isolated from the step 1 test. Then, the order book is asserted to ensure that it has only the manually generated orders from before. The step 1 function is called and the ask order which is created by the actor is purchased through a manually generated bid order. Step 2 then begins and there are three alternatives for what happens next. In the first case, the lowest ask order that is initially read is still in the order book, therefore the generated bid order from the actor executes immediately and fills the ask. In this case, the generation of the bid order is tested by printing the order book and verifying that the lowest ask order has its volume reduced by the amount that the actor purchased. The last step tests the final balance of the actor and the data collection by printing all the values. In the second alternative, the bid order from the actor does not fill an ask order immediately. This is tested by removing the lowest ask and then placing the bid and printing the order book to ensure that the bid order shows up in the order book; it has not filled any asks. In this case, the timing out of orders is tested by ensuring that the bid order expires and no longer appears in the order book. In the last alternative, the bid order is partially filled, but not completely. This is tested by removing the lowest ask order and manually generating one with the same rate, but a lower volume. Printing the order book then shows that the bid order still exists and is partially filled.

3.4 Data Collection

In order to produce results to analyze the algorithm, data is collected about the algorithm and the simulation it runs in. Alongside the implementation of the algorithm itself, the simulation is developed to collect and record data during the course of operation. In this way, information such as the number of transactions completed during a run of the simulation can be accessed and used for analysis.

Two types of data are collected for two different purposes. Firstly, the account balance for the arbitrage actor is recorded each time the algorithm changes state. This is used as a secondary form of verification that the software is working as intended; the changes to the balance at each state are shown in table 3.1.

Table 3.1: Changes to account balance at each state

State	Change to account balance*
START_TRANSACTION	Balance remains same
START_STEP_ONE	Base asset amount decreases by invest amount. Core asset amount decreases by fee.
WAIT_STEP_ONE	Core asset amount increases by amount it is partially filled.
EXPIRED_STEP_ONE	Base asset amount increases by invest amount. Core amount increases by fee.
COMPLETE_STEP_ONE	Core asset amount increases by step 1 order amount.
START_STEP_TWO	Core asset amount decreases by fee and amount purchased from step 1 order.
WAIT_STEP_TWO	Target asset amount increases by partially purchased amount.
EXPIRED_STEP_TWO	Core asset amount increases by fee and volume of the bid minus the amount that is partially sold.
COMPLETE_STEP_TWO	Target asset amount increases by step 2 order amount.
COMPLETE_TRANSACTION	Balance remains same
STUCK	Balance remains same

*Due to restrictions in the current form of the blockchain software, a fee is required every time an order is placed. Since there should be no fees for trading, this is solved by giving the actor the amount of core needed to make all the trades he makes during the simulation (4 million).

The second type of data collected by the algorithm is the data about each transaction. This is used to rate the performance of the algorithm. The transaction data is as follows.

- Transaction Number
- Target Asset
- Current State
- Amount of Base Sold
- Amount of Core Traded
- Amount of Target Bought
- Amount of Core - Partially Filled
- Amount of Target - Partially Filled
- Expected Revenue - Start
- Expected Revenue - End
- Live Forex Rate - Start
- Live Forex Rate - End
- Step 1 Ask Rate
- Step 2 Bid Rate

3.5 Assessing Validity of the Data Collected

To ensure meaningful results, the data collected is assessed for its validity. For the data that is collected on the actor's account balance, the validity is ensured by performing a hand calculation of the account balance and following the changes noted in table 3.1. In addition, the validity of the individual java function call to the blockchain API to retrieve the account balance is ensured by the external sponsor. Test functions are available to test and assert the proper functionality of the API.

The simulation is stochastic in that the trading intentions and the pricing strategy use random number generators in part of their execution. Due to the inherent randomness in the simulation because of this, the simulation is run 50 times. This ensures that the variance due to the randomness does not have a significant effect on the results. Furthermore, this ensures the validity of the transaction data that is used to rate the performance of the algorithm; average values are used for the performance ratings over 50 runs.

The only component of the simulation that is not explicitly validated by this research is how closely the given trading intentions represent traders' behavior on the real market. This is inconsequential however, because the algorithm

seeks only to detect arbitrage by identifying price discrepancies between the two markets. Therefore, even if the simulated traders do not act exactly as real traders do, the software is still verified by acting only when there are opportunities for arbitrage.

3.6 Planned Data Analysis

For both types of data collected, account balances and transaction data, the raw data are cleaned using java. When the data are saved to output files, they are saved as either text files or comma-separated value (CSV) files. For each run of the simulation, three text files are created. One holds all the text that is printed to the output console during the simulation. The second text file holds the details of each of the trading intentions for the simulation. The last one holds all of the simulation parameters. In addition to the text files, CSV files are generated from the simulation. There is only one copy of each CSV file however; each run appends data to the same file rather than making a new one. The following are recorded in CSV files.

- Forex data
  - Transaction number
  - Target asset
  - Live forex market rate at start
  - Live forex market rate at end
- Risk data
  - Simulation number
  - Number of total transactions
  - Number of completed transactions
  - Number of transactions expired at step 1
  - Number of transactions expired at step 2
  - Total amount of leftover CORE from expired step 2 orders
- Profit data
  - Simulation number
  - Transaction number
  - Profit for each transaction

- Expected profit at start
- Difference from actual to expected profit

To allow for reproducibility, the forex data is needed. Also, the simulation parameters and trading intentions are needed. With the forex data CSV file, and these two text files, it is possible to reproduce a specific run of the simulation exactly.

To rate the performance of the algorithm, the risk and profit data are needed. The first performance measure of the algorithm is profit. The profit is calculated for each transaction. These values are averaged over the total number of completed transactions for all 50 runs. The total average profit is given. In addition, for each transaction, the difference in the actual profit and the expected profit at the beginning of the transaction is calculated. The proportion of times the expected profit at the beginning is greater than the actual profit is determined over all runs. This value represents a percentage of the times actual profit is less than expected.

The second measure of performance of the algorithm is risk. The risk is measured by the total number of transactions where a loss occurs. This is used to form a proportion of transactions that make a loss over the total number of transactions. Also, a measure is made of the total number and proportion of transactions that expire during step 2 since these transactions lead to leftover CORE.

3.7 Evaluation framework

Evaluation of the research is split into two categories. Firstly, the software is evaluated to determine whether or not it is functional. Secondly, the performance of the algorithm is determined to see how effective it is as a trading strategy. While the software is the main focus of the research, measuring the performance of the algorithm could lead to interesting results.

The software evaluation is done easily since test-driven development is used as the development practice. If the software passes all of the tests, then it is fully functional. With the goal of the project: “to provide an upgraded, robust piece of software that acts as a trading simulator capable of using the developed arbitrage

trading strategy," the software system will be considered successfully developed if it accomplishes this.

To evaluate the performance of the algorithm, there are four metrics to use: average profit for all transactions, number of transactions where expected profit exceeds actual profit, number of transactions with negative profit, and number of transactions that expire during step 2. However, since the focus of the work is the development and implementation of the algorithm, the risk and profitability measures are not evaluated. The performance measure are simply given, but not evaluated. It is up to the readers of this thesis to evaluate these measures and determine the usefulness of this algorithm in practice.

4 Design and Implementation

The software described here is a combination of brand new features and components added to the sponsor's system as well as continued development on existing parts. The pricing strategy for gateways is explained in the first subsection of the chapter. After that, the trading algorithm is described in detail.

4.1 Pricing Strategy for Gateways in the Trading Simulation

The sponsor provided a simulation program which allows for trading between different actors, called gateways, on the cryptocurrency market who deal in their own currencies (which represent fiat currencies) and the core cryptocurrency. To create the environment on which to run the arbitrage trading algorithm, this simulation program is expanded on and developed further.

One of the key components of the simulation is the pricing strategy which each actor on the market uses to determine how to price the ask orders they place on the market. Since the provided simulation program only has simple pricing strategies where each actor uses a constant rate for every trade (1 to 1 for every currency pair), a new pricing strategy is added which gives more accurate price rates. The advantage of developing a new pricing strategy which gives accurate pricing data is that the simulated market much more closely represents the real market; this facilitates finding arbitrage opportunities possible. In addition, a more realistic pricing strategy helps to validate the usefulness of the trading algorithm since the simulated market more closely represents the real market.

The newly developed pricing strategy works in the trading simulation by using live pricing data from the BitShares (BTS) cryptocurrency market. The process is illustrated in Figure 4.1. To begin, the simulation starts and runs the setup. During setup, the BitShares market is queried by making use of its API. The order books for the following currency pairs are retrieved: BitUSD/BTS, BitEUR/BTS, and BitCNY/BTS. The currency, BTS, is the core cryptocurrency on the BitShares market; the core cryptocurrency on the sponsor's blockchain system is represented by BTS. From these order books, the exchange rates are gathered

for each currency pair and used to determine the prices at which gateways will create ask orders to sell their own currency. There is a one-to-one correlation for each of three virtual "Bit" currencies and the gateway's own created currencies (e.g. gateway A's currency is represented by BitUSD, gateway B's currency is represented by BitEUR, gateway C's currency is represented by BitCNY).

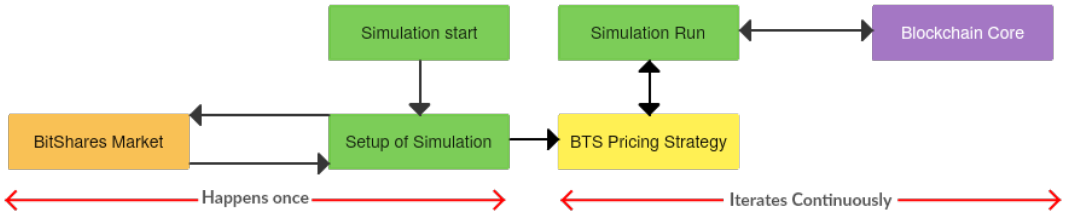

Figure 4.1: *Simulation Diagram with Pricing Strategy*

The base price rate, P_b is determined by taking the middle limit order from the order book; this gives the mid-price for the currency pair. With P_b, a normal distribution is created with 0 as the mean and 5% of this value as the standard deviation: $N(0, 0.05 \times P_b)$. The price rates that the gateways use during the simulation to create a price for their limit orders are denoted P_g and are determined by adding a sample from the normal distribution to the base price.

$$P_g = P_b + sample(N(0, 0.05 \times P_b)) \quad (1)$$

A normal distribution around the price rates from the BitShares market is used as a method for simulating trader's pricing behavior on the real market. This method of using a normal distribution for simulating pricing is described in Teng Jiang's paper, "An Agent-Based Simulation of Trader Behaviour..." She explains that, in her simulation of traders on the stock market, a normal distribution of the stock price can be used by the traders to form their trading prices. [19] Since the situations are very similar - determining which price a trader uses in a market simulation - this methodology is applied for the developed pricing strategy. Jiang also uses a standard deviation of 5% of the starting value during the experimental setup. The starting value is 100 pence and standard deviation 5 pence in the parameters for her experiment. [19]

4.2 Automated Triangular Arbitrage Trading Strategy

This section discusses the design and implementation of the main piece of the work - the arbitrage trading algorithm. The algorithm runs inside the simulation given by the sponsor, with the addition of the pricing strategy described above. A general overview of the algorithm is given by Figure 4.2. It is discussed in detail in the following subsections.

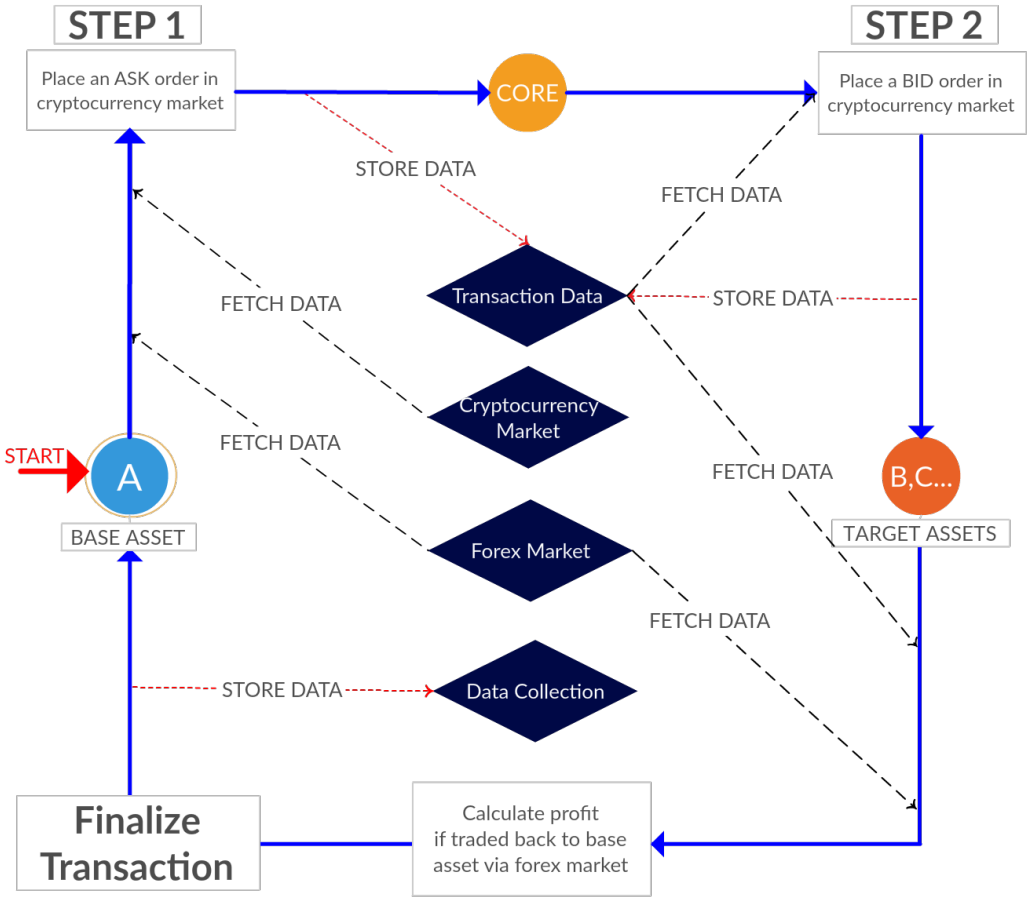

Figure 4.2: *Arbitrage Algorithm Overview Diagram*

The arbitrage trading algorithm seeks to detect arbitrage opportunities between the live forex market and the cryptocurrency market and trade based on these opportunities to make the user a profit. To use the algorithm, the trader selects one fiat currency as his base currency, and one or more fiat currencies as targets. The base currency is the one which the trader wants to profit in. The target currencies are those that the trader will attempt to purchase and then exchange back into the base currency.

The overall concept of the algorithm is for the trader to profit by using triangular arbitrage between the live and crypto markets. Using the base and target currencies selected, the algorithm first identifies an arbitrage opportunity between the rate at which those currencies are exchanged on the two different markets. The algorithm only looks for arbitrage where the rate is lower on the crypto market than on the live forex market; a profit is made by buying the target currency through the crypto market and then buying back the base currency through the live market. Arbitrage is not detected and used in the other direction (i.e. buying target through the live market and then buying base back through the crypto market) because one of the fallbacks of the live market is that it takes two days for money to be transferred between currencies, and in that time the rates may change significantly on the crypto market. Also, because of this transfer delay, the trader must wait to receive his profits after a successful run of the algorithm. Therefore, the algorithm runs in the simulation until either the simulation ends, or the actor's balance of base currency is zero; i.e. he would not be able to use his profits to reinvest immediately.

To develop the algorithm in a modularized manner, it is broken down into three major steps: step 1, step 2, and finalizing the transaction. After an arbitrage opportunity is identified, step 1 occurs and the algorithm places an ask order to trade the trader's base currency for the core cryptocurrency. Next, step 2 occurs and the algorithm places a bid order to trade the core for the target cryptocurrency. Finally, the algorithm finalizes the transaction by recording all necessary data and calculating the expected profit if the trader converts back to base through the live market. Since the algorithm only runs in a simulation and not the real market, the actual purchase of the base currency through the live forex market cannot be made, therefore the expected profit is only calculated at the end rather than being explicitly shown through an order.

4.2.1 Automation and Algorithm States

While running in the simulation, the algorithm changes from state to state. Automation of the algorithm is guaranteed by the developed state system. The different states of the algorithm and the behaviour of the algorithm for each round

of the simulation are shown in figure 4.3.

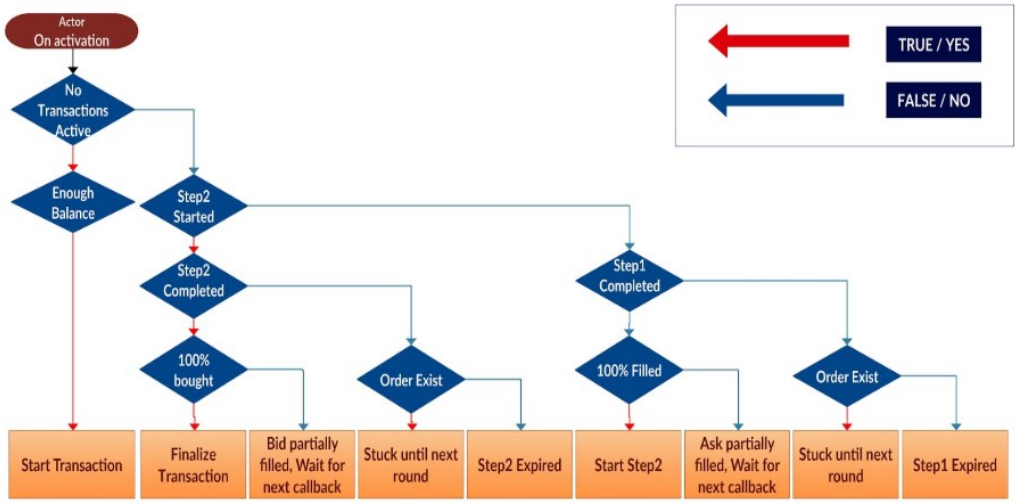

Figure 4.3: *Algorithm States: onActivation*

The *onActivation* function is called at the beginning of each round of the simulation. It starts by checking to see if there is a transaction currently active, and begins one if not. Only one transaction occurs at a time. The reason for this is two fold; firstly, because it only uses one base currency, the algorithm would compete with itself by placing multiple ask orders at the same time. Also, since the actor never replenishes his balance of base currency during the simulation, a predetermined investment amount is chosen. This amount is 25% of his starting balance of base currency. The investment amount is limited so that the actor does not use his entire balance in one transaction; spreading the investment over multiple transactions allows the actor to identify more profitable opportunities over the course of the simulation.

Each of the rectangles at the bottom of figure 4.3 represents the different states of the algorithms. The rest of the diagram display the logic used to direct the algorithm on what to do in each state and how to navigate between them. Step 1 and step 2 each have 4 states: start, wait (while an order is partially filled), completed, and expired. Also, the algorithm enters the stuck state when waiting for a callback function to be called from a market notification subscription. Finally, there are two states for the transaction as a whole: start and completed.

4.2.2 Detecting Arbitrage and Determining Price Rates

To begin with, the algorithm seeks to detect arbitrage opportunities by comparing price rates from the live forex market and the cryptocurrency market. In the following description, the exchange between the United States dollar and the euro will be used as an example, but any other currency pair that is available on both markets would work as well. The actor running the arbitrage algorithm uses the dollar as his base currency and the euro as his target.

Firstly, the algorithm queries the live forex market and receives the current exchange rate available for traders for EUR to USD. This rate is expressed as R_L. This query is made to the REST API for the live market, which works by accepting the query as an HTTP request and then returning the exchange rate in the same manner.

Next, the algorithm makes a request to the API for the blockchain and retrieves the order book for the market between the core cryptocurrency, CORE, and the cryptocurrency represented by BitEUR. Using this order book, the current lowest ask price for this currency pair is found and is denoted as R_B. Before completing step 1 of the algorithm, the order book for the market between CORE and the crypto represented by BitUSD is also retrieved and the lowest ask price is determined, R_A. Over the course of the simulation, the algorithm continuously makes these API calls and updates the rates R_L, R_B, and R_A. This is done with the intention that, at any given time, when a price rate is needed and accessed from the algorithm, the most recent rate is available.

4.2.3 Step 1

After all of the necessary price rates are determined from the API calls to the live forex market and the cryptocurrency market, the arbitrage actor determines at what rate he will place an ask order for core. The calculated ask rate for the actor to break even on a trade is denoted by A. First, the value is generated by multiplying R_L and the inverse of R_B and then inverting the result.

$$A = \frac{1}{R_L \times \frac{1}{R_B}} \tag{2}$$

This result for rate A is shown by using the units for each rate in the equation. Rate A is expressed in units of CORE per BitUSD; in step 1, the actor seeks to exchange his own BitUSD for CORE.

$$\frac{CORE}{BitUSD} = \frac{1}{\frac{BitUSD}{BitEUR} \times \frac{1}{\frac{CORE}{BitEUR}}} \tag{3}$$

$$\vdots$$

$$\frac{CORE}{BitUSD} = \frac{CORE}{BitUSD} \tag{4}$$

After using the formula above to determine rate A, the value is compared with R_A and is checked to see if it is greater or less than the rate from the order book of BitUSD to CORE orders. Since the rate A is the rate at which the actor would break even on a trade, if A is less than R_A, then a profitable arbitrage opportunity is identified. At this point, an optimization of the algorithm also occurs - since the rate A is calculated for each target currency, the lowest one is chosen (and thus the specific target is chosen) since this one will be the most profitable. Now, the rate at which the actor places his ask order is set at R_A, and since this is the lowest ask rate in the order book, this order will be filled before other asks. The larger the difference between A and R_A, the more profit the actor will earn, since he will trade the same amount of BitUSD for comparatively more CORE than the amount that would have make him break even.

4.2.4 Step 2

Once step 1 of the algorithm is complete, the actor has a balance of CORE from trading his BitUSD. To complete step 2, he trades this CORE for BitEUR. Using the lowest ask rate from the order book of orders from CORE to BitEUR - the rate R_B - the actor places a bid order with all the CORE earned from step 1. If this bid order fills an ask order successfully, then the actor receives BitEUR and moves on to finalizing the transaction. If the bid order is unsuccessful however, and it does not fill an ask order in a predetermined amount of time, then the bid order times out. In this case, the transaction is marked as incomplete, and the algorithm returns to the beginning.

Although many definitions of arbitrage describe the trading strategy as 'risk-free' profit, this algorithm does include some risk. In the case where the step 2 bid order times out and the whole transaction is marked as incomplete, the actor is left with leftover CORE and does not make use of this for the rest of the simulation. This is done so that at the end of the simulation, the extra CORE can be summed and used as a secondary method of verifying whether a transaction is complete or not. The amount of risk, and its effect on the overall profitability of the algorithm is discussed further in the final chapter of this text.

The risk of not completing the step 2 bid order exists because of the fact that it may take some time for the step 1 ask order to be filled. During that time, the lowest ask rate, R_B, that was used to calculate the rate at which the actor placed his step 1 order (to break even) may no longer be the current lowest ask rate between BitEUR and CORE. This occurs when the ask orders at that rate are filled and only orders with higher ask rates remain. By the time step 1 completes and the actor begins step 2, there are only ask orders with higher rates than the rate with which the actor makes his bid order for BitEUR.

4.2.5 Finalizing Transaction and Data Collection

During step 1 and step 2, the program records updated and used data. Once step 1 begins, the simulation stores: the retrieved rate from the forex market, the placed ask rate, the threshold bid rate for step 2, the updated state of the algorithm, and more. These data are used for future activations such as placing the bid order in step 2. Moreover, the algorithm finalizes the data collection of the transaction data after the execution of step 2. It saves the amount of target currency bought. This amount has the potential to be greater than is first expected; the ask rates for the 'core to target' market could have decreased in the time it takes to complete step 1; this allows more target currency to be purchased with the same amount of core.

In the case where step 1 or step 2 expires due to an order timing out, the algorithm tracks the updated balances so the next transaction can be synchronized. Furthermore, the algorithm records its transaction state as expired. Lastly, once the simulation completes, it exports the output to a text file. In

addition, the algorithm exports the simulation parameters, the trading intentions for the gateways, the forex market rates, and other pieces of information into different files.

5 Results and Analysis

To determine the effectiveness of the developed software system and whether or not the goals are met for this research, the results of the work are generated and analyzed. The results are generated from three sources: the unit tests, the actor's account balance throughout the simulation, and the transactions that occur. From these sources, data is gathered and presented in the first subsection: Major Results. There are two main results that are generated and analyzed. First, there are results on the software system itself which are composed of the unit tests and the actor's account balance. The second set of results regard the performance of the algorithm and they are composed of four different metrics:

- total average profit for each transaction,
- the number and proportion of transactions where the actual profit is less than the expected profit at the start,
- the number and proportion of transactions that result in a loss (i.e. negative profit),
- the number and proportion of transactions that expire during step 2.

Along with presenting the results, an analysis on the validity of them is given. The chapter concludes with a discussion on the results.

5.1 Major Results

To generate these results, the simulation is run 50 times. The results are gathered as outputs and displayed here.

5.1.1 Unit Tests

The unit tests for the software system determine whether or not the system behaves correctly, according to the design. The design of the algorithm and the pricing strategy for the traders in the simulation is made based on the goal outlined in the introduction. The unit tests, therefore test and validate that the software

system is successful and works as intended. The name and result of each test is given below in table 5.1.

Table 5.1: Unit Tests

Test Name	**Result**
Pricing Strategy	Pass
Cryptocurrency Market Fetch	Pass
Random Generator	Pass
Lowest Ask Rate from Order Book	Pass
Lowest Ask Rate from Live Market	Pass
Update Price Rate	Pass
Determine Price Rate	Pass
Live Market Fetch	Pass
Live Market Configuration	Pass
Price Polling	Pass
Step 1 Case 1	Pass
Step 1 Case 2	Pass
Step 1 Case 3	Pass
Step 2 Case 1	Pass
Step 2 Case 2	Pass
Step 2 Case 3	Pass

5.1.2 Account Balance

The actor's account balance is printed out during each state change of the algorithm. This is done as a secondary form of verification that the software functions as intended. Table 5.2 displays a sample output of the actor's account balance during one run of the simulation. The table contains three transactions.

Table 5.2: Sample Account Balance

#	Current State	Target	A Balance	B Balance	C Balance	Core Balance
4	START_STEP_ONE	C	750000	0	1720733	3000000
4	STUCK	C	500000	0	1720733	31666167
4	COMPLETE_STEP_ONE	C	500000	0	1720733	31666167
4	START_STEP_TWO	C	500000	0	1720733	31666167
4	EXPIRED_STEP_TWO	C	500000	0	1720733	31666167
5	START_STEP_ONE	C	500000	0	1720733	31666167
5	STUCK	C	250000	0	1720733	59188413
5	COMPLETE_STEP_ONE	C	250000	0	1720733	59188413
5	START_STEP_TWO	C	250000	0	1720733	59188413
5	EXPIRED_STEP_TWO	C	250000	0	1720733	59188413
*	⋮					
0	START_STEP_ONE	C	1000000	0	0	4000000
0	COMPLETE_STEP_ONE	C	750000	0	0	30953955
0	START_STEP_TWO	C	750000	0	0	30953955
0	COMPLETE_STEP_TWO	C	750000	0	1656782	3000004
0	COMPLETE_TRANSACTION	C	750000	0	1656782	3000004
1	START_STEP_ONE	C	750000	0	1656782	3000004
1	STUCK	C	500000	0	1656782	2500004
1	EXPIRED_STEP_ONE	C	750000	0	1656782	3000004

*The row marked with a star indicates the end of one simulation and the start of another. This explains the balance change and the different transaction numbers.table

5.1.3 Transaction Data - Performance Measurements

The four performance metrics are made using data gathered throughout the course of the algorithm. Three of the metrics are the average profit earned by the actor over all completed transactions, the difference in actual and expected profit, and the transactions which result in a loss. The profit (actual and expected) earned by the actor is shown for the first 16 transactions in table 5.3. Only 16 transactions are selected as a sample for brevity's sake.

Table 5.3: Sample Performance Measurements (1)

Transactions	Actual Profit	Expected profit at start of transaction	Difference from actual to expected profit
0	18316	18316	0
1	33721	-1	33722
2	-1	-1	0
3	0	-1	1
4	31001	-1	31002
5	1769	-1	1770
6	17116	7381	9735
7	-1	-1	0
8	2109	-1	2110
9	53947	34772	19175
10	82709	39909	42800
11	26024	-1	26025
12	21491	6682	14809
13	21244	14297	6947
14	6330	6330	0
15	9037	-1	9038

The final performance metric: transactions which expire during step 2 is measured by the transaction data. The first 16 transactions are displayed in table 5.4.

Table 5.4: Sample Performance Measurements (2)

Simulations	Total transactions	Number of completed transactions	Number of step 1 expired	Number of step 2 expired
0	5	2	2	1
1	3	3	0	0
2	4	2	1	1
3	13	0	13	0
4	4	3	1	0
5	4	3	1	0
6	5	2	2	1
7	4	2	1	1
8	4	3	1	0
9	3	3	0	0
10	3	3	0	0
11	12	1	11	0
12	3	3	0	0
13	4	2	1	1
14	4	3	1	0
15	3	3	0	0

5.2 Analysis

5.2.1 Software

To validate the software, two results are analyzed: unit tests and the actor's account balance. The unit tests are the primary form of validation since they

validate the implementation of the algorithm. By monitoring the actor's account balance throughout the simulation, the design of the algorithm is validated. Having the correct balance throughout the simulation also acts as a secondary form of validation for the implementation.

As can be seen in table 5.1, the software passes all the unit tests. Because of this, the software is validated according to the design and intended implementation.

In table 5.2, the actor's account balance is shown for four transactions. Each transaction begins with the state START_STEP_ONE and ends with either an expiration or COMPLETE_TRANSACTION. In this sample, by the time the first transaction that is shown occurs, the actor has already made a transaction and therefore has some currency C and has spent A and core. At the beginning of transaction number 4, the actor holds with the correct starting values: 750,000 units of A (1,000,000 minus the invest amount from the previous transaction 250,000) and 3,000,000 units of core (4,000,000 minus the two trading fees from the previous, 500,000 each). Then, as step one completes, the actor trades 250,000 units of A and earns core in return. The actor then starts step two and places the bid order, but the order expires and therefore the actor is left with the balance of 500,000 units of A and 31,666,167 units of core.

Transaction number 5 begins in the same way as the previous transaction. The ask order is successfully filled in this transaction, and in the state COMPLETE_STEP_ONE the actor spends another 250,000 units of A and receives 28,022,246 units of core. Next, the actor places the bid order for step 2 of the algorithm. After the state START_STEP_TWO, since the bid order does not fill an ask order before the timeout, the order expires and the actor is left with the same balance. This is reflected in the actor's balance of core of 59,188,413 in EXPIRED_STEP_TWO.

In transaction number 0, the actor begins the algorithm again and correctly holds the starting values for each currency; 1,000,000 units of A, 0 B, 0 C, and 4,000,000 units of core. He starts step 1 and places an ask order. He spends 250,000 units of A and pays the fee of 500,000 units of core. The ask order is filled and step 1 completes so the actor receives 27,453,955 units of core to increase his core balance to 30,953,955. The actor then begins step 2 and places a bid

order for currency C, which is filled. After step 2 completes, the actor correctly loses 500,000 units of core for the trading fee and the 27,453,955 units of core he received from step 1. In addition, his balance of currency C increases by 1,656,782 from the step 2 order being filled. The actor has a final core balance of 3,000,004; he has 4 extra units of core since the bid order in step 2 filled an ask order with a lower rate, therefore the actor received more of currency C than first expected. Because the order amount is rounded to the nearest integer of currency C, the actor is left a few extra units of core.

The final transaction, number 1, the actor attempts another trade and starts step 1. He pays the trading fee of 500,000 units of core and the invest amount of 250,000 units of A. Since the order expires however, the actor is returned the amounts of A and core.

By passing all of the unit tests and having the correct amount of each currency during each state of the algorithm, the software is validated. Both the design and the implementation of the algorithm are validated in accomplishing what they are intended to do; the software system works and can run to produce results. Furthermore, the algorithm correctly identifies arbitrage opportunities and makes trades in an attempt to profit from them.

5.2.2 Performance Measurements

The performance measurements are analyzed in a graphical format. Figure 5.1 displays the profit for each transaction with the average profit over all transactions shown as a dotted line. Figure 5.2 shows the second performance measurement: the difference between actual profit and expected profit at the beginning of the transaction. The transactions are shown in figure 5.3 and categorized into those which earn a profit, a loss, and those that break even on the original investment. The final performance measure is visualized in figure 5.4, which shows the number of transactions that expire during step 2 for each run of the simulation.

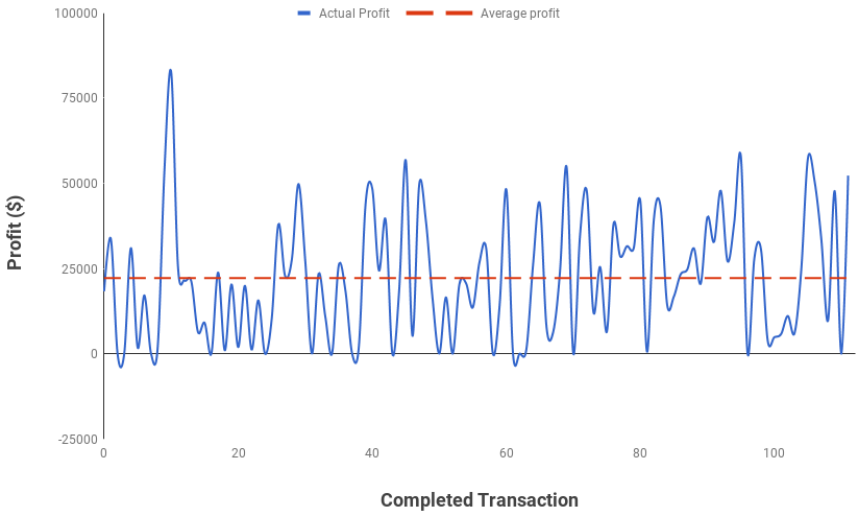

Figure 5.1: *Profit per transaction with average*

The profit is measured in units of currency A. The highest profit earned in one transaction is 82,709, although this value is an outlier. The lowest profit earned in one transaction (the biggest loss) is -16. The total average profit over every transaction is 22,188.5625. Since the invest amount is constant at 250,000 units, this equates to an average profit of about 8.9%.

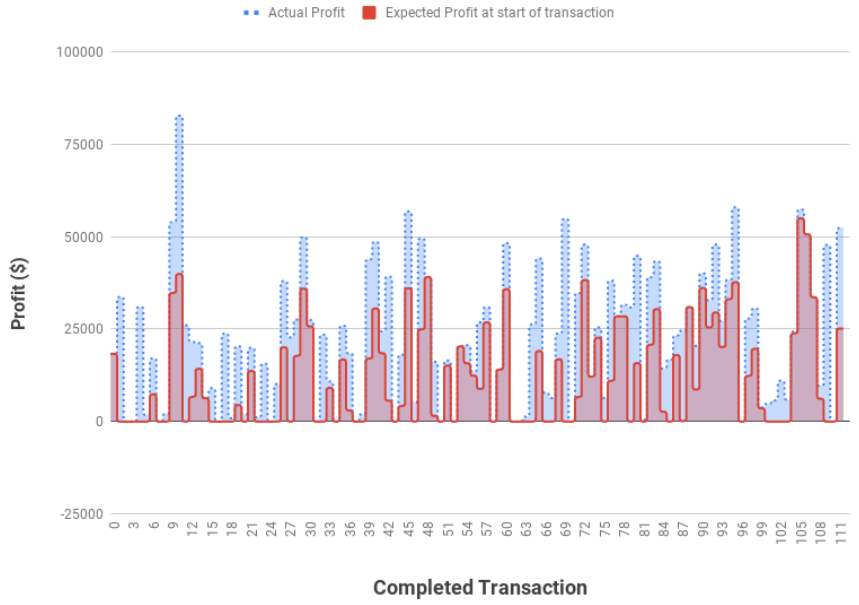

Figure 5.2: *Profit per transaction: Expected vs Actual*

The total number of transactions where the expected profit at the start is greater than the actual profit is 2. With a total of 112 completed transactions, this results in a proportion of about 1.8% of transactions where the actual profit is lower than expected.

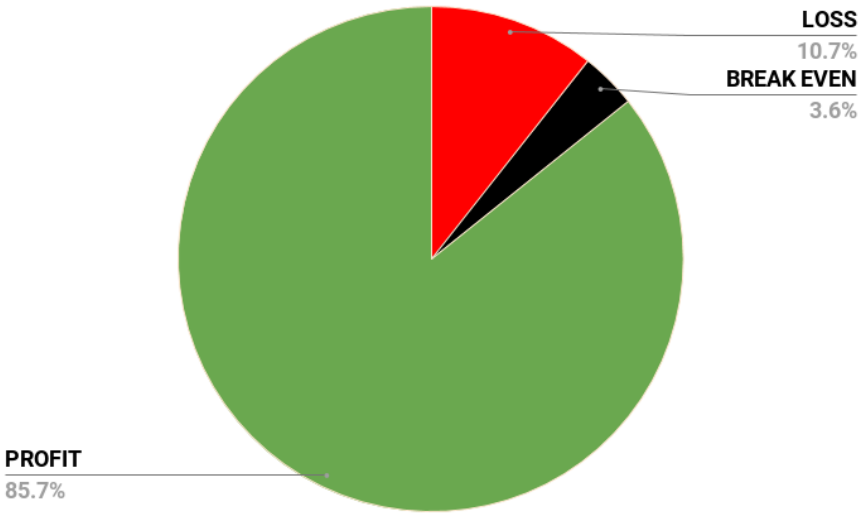

Figure 5.3: *Transactions: profit, loss, and break even*

The total number of completed transactions that result in a loss is 12. The total number of completed transactions that break even on the original investment is 4. The remaining 96 completed transactions result in a positive profit. The proportion of completed transactions which make a loss is about 10.7%. Out of the 12 transactions that result in a loss, 11 of them report a loss of -1. In addition, there are some transactions in table 5.3 that display an expected profit of -1. Both of these results are due to a rounding error since the trades can only be made in integer values of currency. The algorithm expected to break even, or make a very small profit on these transactions, but due to a rounding error, the expected or actual profit becomes -1.

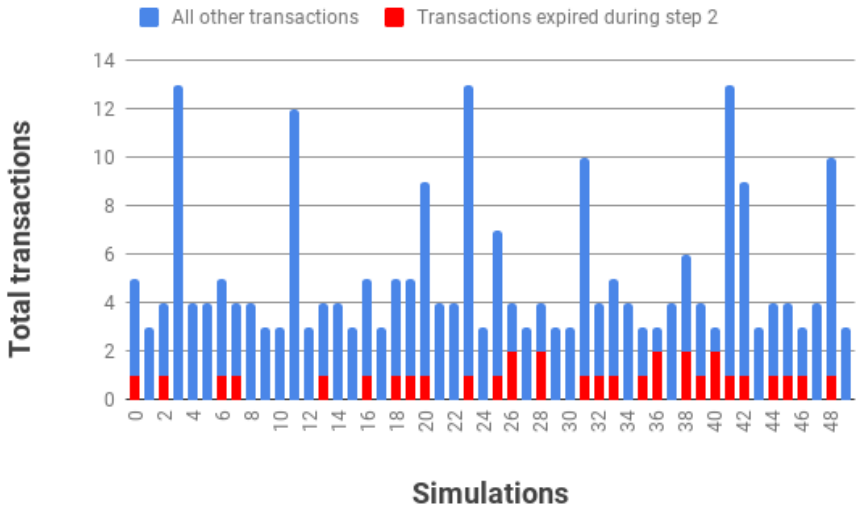

Figure 5.4: *Transactions which expire during step 2*

The total number of transactions that expire during step 2 is 32. The total number of transactions (complete and incomplete) is 254. The proportion of transactions which expire during step 2 out of the total is about 12.6%.

5.3 Discussion

With the goal of this research being: given a trading simulation, design and implement an arbitrage trading algorithm which can identify arbitrage opportunities and attempt to make profit by trading based on those opportunities, we consider these results a success. The success of the unit tests and, in general, the practice of test-driven development affirm that the design and development of the algorithm is a success.

While the particular financial strength of the algorithm is not analyzed in depth, we consider the overall performance of the algorithm satisfactory. It is not perfect, since there is risk when running the algorithm. Out of 112 transactions which completed, 12 of them resulted in negative profit, meaning that user loses money in this case. Of those 12, however, 11 of them made a loss of exactly 1 unit of currency. This is caused due to the values being rounded during the algorithm. If these are considered as breaking even, then only one transaction results in a significant loss. This brings the proportion of transactions that make a loss down

to below 1%. This value is very small, but the risk of losing money still exists.

In addition to making a loss, the algorithm also carries risk of expiring during step 2. When this occurs, the core earned during step 1 is held and not used for the rest of the simulation. In reality, this could potentially be used to purchase more of the target asset or even buy back the base asset. Since an expiration during step 2 occurs if the rate that would guarantee profit is not met however, then it is an inherent risk of the algorithm that the user could be stuck with his investment in core currency.

Although the algorithm carries some risk, it also tends to make a profit. On average, the profit over all completed transactions is about 8.9% of the investment amount. Assuming that a potential user of the algorithm can make many investments and use the algorithm many times, he will tend to make money. In this way, the design of the algorithm is successful.

6 Conclusions and Future Work

6.1 Conclusions

This project brings together knowledge and challenges from two separate industries, finance and technology, in a novel way. The strictly financial side of the algorithm, the triangular arbitrage, is already a well known and established trading tool. The implementation of a trading strategy into software is also a well known practice. The introduction of the triangular arbitrage algorithm into technology, specifically for use with a blockchain however, is a new idea. The combination of these two well known practices make a unique software product which definitely warrants further research.

In the introduction of this thesis, the problem is identified mainly as three technical challenges: implementing a pricing strategy, implementing the trading algorithm, and developing the given trading simulation. All three of these challenges were approached using the practice of test-driven development. Overall, this was the most challenging part of the work. Using test-driven development proved to be very effective, but difficult. It is very beneficial since it makes the verification of the software simple and easy to do. On the other hand, it is time consuming to write the tests and at times seems unnecessary. In the end, the benefits of using test-driven development outweigh the costs though, and we recommend this practice for any software development.

Although this project investigates topics from two separate industries, the main focus of the work is the technical side. The purpose and goal in the introduction chapter of this project reflect this as well. While the financial side is definitely interesting and worthy of further studies, the trading strategy itself is used as a basis for research using software development. In that sense, this project is a clear success. This research topic expanded our knowledge of APIs, blockchain, cryptocurrencies, software development practices, and many more technical items. Overall, this work is very well rounded within the technical topics that it covers and it encourages us to research further into the software industry.

Overall, we feel that we have met our goals and have gained valuable insights by doing this project. Firstly, working with blockchain technology has inspired

both of us to study this topic further. Blockchain is such a diverse and dynamic technology and it will have uses in the future that have not even been surmised today. Being able to work with this in the research project is very beneficial since blockchain technology will be around for years to come and will become increasingly important in the software industry. In addition to blockchain, our knowledge of the financial world has greatly increased. The industry that blends finance and technology, appropriately referred to as fintech, is another growing industry. As technology continually advances and more and more automation is implemented in society, this project has given us a valuable experience. We have been able to work on automating an existing idea, i.e. triangular arbitrage, using software. Since this will surely continue in the future, we are glad to have experienced this process firsthand.

If we were to do this project over again, one significant step we would implement that we did not, would be to clearly define the software requirements before beginning to program. In the practical world of software development, these requirements are often created by the stakeholders and the developers who communicate with each other. Since this is a research project however, we did not define these ourselves in the beginning of the work, but we have come to realize that this would have been beneficial. Having clearly defined requirements makes the process of defining the goal and methodology for the project easier. To any others working in this area, we highly recommend making clear software requirements before beginning the implementation.

6.2 Limitations

The main limitations we experienced in this project were related to the financial aspects of the work, but there is also one main technical limitation: software requirements. As the software development process is an iterative one, it is always a great benefit if one can reduce the time taken to do any particular task. Saving time means that one has more time to improve the software through iterations of testing and optimizing components of it. In conclusion, we feel that if we had taken the time at the beginning of the work to make well-defined software requirements for our system, we would have saved time overall. Having clear

requirements means less time spent making decisions during the time where we were focused on implementation. Since the project is completed in a limited time frame, this would have given us more time to improve the software after the bulk of the implementation was complete.

The financial side of the project was challenging in that it took a lot of research in the beginning to get our financial knowledge up to the level that was required to understand many parts of the system. We researched the topic of cryptocurrency, for example, a lot in order to understand how it attains its value, since it does not have an inherent value. Also, if we had a greater financial knowledge and education in the industry, we could analyze the performance of the algorithm in a more comprehensive manner. The final difficulty from the financial side of the work was making the simulation to replicate the market well. Since our knowledge in this area is limited, we likely could have made further improvements to the simulation if we understood financial markets better.

6.3 Future Work

This project is only the beginning to a very interesting field of work; fintech. Implementing this trading algorithm in software has inspired us to continue research in the field, as we hope it will do to others as well. To continue this work, there are a few obvious things that could be done next. Firstly, for all of the implementation done in this project, the blockchain system was run locally. An improvement can be made here by running the system on a server and allowing others to connect concurrently, so that the algorithm can be run on a system with real traders. This improvement can be taken even further by testing the algorithm with real funds on a real market, rather than just in a simulation.

Another significant improvement that can be made to the algorithm is using buffer accounts for each different currency type. This could be implemented by having the user control an account with each currency type that he will trade, and start with an initial balance in each account. Then, he could perform each step of the algorithm concurrently without having to wait for his investment to travel through the system. Instead, he would just fill and drain the buffer accounts appropriately by making each trade simultaneously. This would even

further reduce the risk of the algorithm, because there would be no situation where the current rate for a trade would change in the time that the trader waits for his money to make its way to the right account. Furthermore, the initial investment amount could be optimized by each trader when using the algorithm. With additional time, we would attempt to review data from the algorithm after a period of use, and analyze this data to optimize the initial investment amount.

A further related topic to the research that we identified, but did not study since it fell out of the scope of this project is being a market maker on one of the specific markets. The cryptocurrency on the sponsor's system acts as a currency basket; it attains its value based on all of the currencies that it can perform trades with. Since the exchange rates between each currency in the basket are known, one could potentially act as a market maker for one market (e.g. USD and core) by providing liquidity to that market and profiting based on a bid/ask spread that he sets himself. This topic presents another opportunity to profit based on trades within the cryptocurrency market using a developed algorithm.

References

[1] Iansiti, M. and Lakhani, K. “The Truth About Blockchain.” In: *Harvard Business Review* (2019). URL: https://hbr.org/2017/01/the-truth-about-blockchain.

[2] *centiglobe web page*. URL: https://www.centiglobe.com/.

[3] W, James and A.Vorontsov. *Test-Driven Development in Microsoft .NET*. URL: https://ptgmedia.pearsoncmg.com/images/9780735619487/samplepages/9780735619487.pdf.

[4] Attreya, Rama. “FOREX Scandal: Top Banks Face Antitrust Fines”. In: *Vol 35 REVIEW. BANKING FIN. LAW. 34* (2015). URL: https://www.bu.edu/rbfl/.

[5] Bradley, Richard. “Blockchain Explained... In Under 100 Words”. In: (2019). URL: https://www2.deloitte.com/ch/en/pages/strategy-operations/articles/blockchain-explained.html.

[6] Singhal B, Dhameja G and P., Panda. “Beginning blockchain. A beginner’s guide to building blockchain solutions”. In: (2018).

[7] *bitcoin website*. URL: https://www.bitcoin.com/.

[8] Szabo, N. “ Smart contracts: building blocks for digital markets.” In: *EXTROPY: The Journal of Transhumanist Thought,(16)* (1996).

[9] “Triennial Survey of foreign exchange and OTC derivatives trading in April 2016.” In: *Bank for International Settlements* (2016). URL: https://www.bis.org/statistics/d11_1.pdf.

[10] Rogoff, K. “The Purchasing Power Parity Puzzle.” In: *Journal of Economic Literature, 34(2) 647-668*. (1996). URL: http://www.jstor.org/stable/2729217.

[11] Kritzer, A. “Forex for beginners a comprehensive guide to profiting from the global currency markets”. In: (2012).

[12] Moosa, I. “ Triangular arbitrage in the spot and forward foreign exchange markets”. In: *Quantitative Finance, 1:4, 387-390* (2001). DOI: DOI:10.1080/713665833.

[13] " Forex - Foreign Currency Transactions". In: (2013). URL: https://www.sec.gov/answers/forcurr.html.

[14] Halls-Moore, M. L. " Successful Algorithmic Trading. Applying the scientific method for profitable trading results". In: (2015).

[15] Dong, H. and Dong, W. " Bitcoin: Exchange rate parity, risk premium, and arbitrage stickiness." In: *British Journal of Economics, Management Trade, 5(1):105–113* (2014).

[16] Pieters, G. and Vivanco, S. " Financial regulations and price inconsistencies across Bitcoin markets." In: *Information Economics and Policy, 39:1–14.* (2017).

[17] Wang J Hsu Y, Chen C. "Triangular no arbitrage Estimation through bitcoin, an application in Venezuelan Bolivars". In: *Asia Pacific Journal of Financial Studies, 47(4), 529-545.* (2018).

[18] Julia, R. et al. "Deviations from Triangular Arbitrage Parity in Foreign Exchange and Bitcoin Markets." In: *Electronic Journal.10.2139 ssrn.3148094.* (2018).

[19] Jiang, Teng. "An Agent-Based Simulation Of Trader Behaviour In Modelling Emergent Phenomena In Stock Market". In: *University College London Msc Financial Computing Project* (2011). URL: http://www.resnovae.org.uk/fccsuclacuk/images/docs/project-resources/Teng-Jiang-2011.pdf.

TRITA-EECS-EX-2019:98